Thai Journal of **Math**ematics
Volume 21 Number 4 (2023)
Pages 785–798

http://thaijmath.in.cmu.ac.th

ISSN 1686-0209

Discrete and Computational Geometry, Graphs, and Games

On the Complexity of Motion Planning Problem of a Forklift

Chao Yang

School of Mathematics and Statistics, Guangdong University of Foreign Studies, Guangzhou 510006, China
e-mail : sokoban2007@163.com

Abstract The robot motion planning problem is one of the most studied problems in computational geometry in continuous settings. We study the Forklift Motion Planning Problem, which is defined based on a puzzle called Zaikoban, in a discrete setting of 3-dimensional space. We demonstrate that the configuration space method can be applied to the discrete case. In particular, we show that the forklift motion planning problem is solvable in polynomial time for the one-box case, and show that the general forklift motion planning problem with multiple boxes is PSPACE-complete by reduction to the nondeterministic constraint logic (NCL) problem.

MSC: 68Q17
Keywords: motion planning; computational complexity; PSPACE-complete

Submission date: 22.01.2022 / Acceptance date: 15.11.2023

1. Introduction

The study of the robot motion planning problem, sometimes referred to as the geometric path planning problem, the generalized mover's problem, or the piano mover's problem, dates back to the 1970s. We refer to [1] for a recent survey of this problem. In the robot motion planning problem, we try to guide a robot from one position to another in a Euclidean space $\mathbb{R}^2$ or $\mathbb{R}^3$ (the workspace), by avoiding the obstacles in the space. Note that in general, the robot may consist of several parts, and each part may be moved independently. A standard way to study the problem is to define the *configuration space*. For example, let the robot be a long rectangle that can slide and rotate in the plane. Then a configuration of this rectangular robot can be exactly described by the coordinates (x, y) of one of its vertices, and the angle θ between its long edge and the x-axis. So the configuration space is the set of all points $(x, y, \theta) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ which are collision-free with all obstacles. Usually, the configuration space has a higher dimension than the workspace.

Published by The Mathematical Association of Thailand.

In general, the continuous version of the robot motion planning problem is PSPACE-hard. Canny gave the first single exponential-time algorithm in the dimensionality of the configuration space in his doctoral thesis [2]. As a corollary, if we fix the dimension of the configuration space, then we will have a polynomial algorithm for the problem.

In real-world applications, there are lots of situations where the robot moves in a discrete manner. But the problem remains PSPACE-complete for the discrete case. The sliding block puzzles can be viewed as a discrete type of motion planning problem, and it is PSPACE-complete, even with blocks of size 1×2 and 2×1. If all blocks are of size 1×1, it becomes a generalization of the Fifteen Puzzle, and the problem is solvable in polynomial time.

The discrete motion planning problems have also been studied by many scholars, see [3–8]. But the workspaces of the models in most of these studies are essentially 2-dimensional. In this paper, we study a 3-dimensional discrete case of the motion planning problem, the motion planning for a forklift in a warehouse. Unlike the sliding block problem which has a workspace of dimension 2, the motion planning problem of a forklift has a workspace of dimension 3. We apply the configuration space method to analyze the problem. As we shall see, the configuration space becomes a graph in the discrete case.

We use a Japanese game called Zaikoban[1] developed by NetFarm in 2007 as a model for the forklift motion planning problem in a warehouse. The game was also available temporarily under the name *Soko Forklift - Zaikoban* for the Android system at Google Play in 2014, and for the iOS system in 2015. See Figure 1 for screenshots taken from the Android version.

FIGURE 1. Screenshots of the game Zaikoban

In the game Zaikoban, a forklift has to transport one or more boxes from their initial positions to their respective goal positions. The warehouse is a 3-dimensional grid with

[1]https://www.netfarm.ne.jp/island/release/070713_01.pdf

walls. The forklift occupies 4 cubes of the grid, and it can move forwards or backwards, turn 90 degrees to the left or right, raise or lower the fork, and load or unload a box from the fork. There are a few additional elements in the game of Zaikoban, such as fragile floors which the forklift can go through only without carrying a box.

Definition 1.1 (Forklift Motion Planning Problem)**.** Given a warehouse with walls, one forklift, one or more boxes with goal positions, and possibly a few additional elements, decide whether the transportation of the boxes can be done by the forklift.

The main contribution of this paper is the following Theorem 1.2 and Theorem 1.3. If there is only one box to be transported, then the configuration space has dimension 4, and there exists a polynomial-time algorithm to find the shortest path.

Theorem 1.2. *The forklift motion planning problem can be solved in polynomial time for one box.*

If there is no limit on the number of boxes, which is equivalent to that there is no limit on the dimensionality of the configuration space, then we show that the problem is PSPACE-complete, by an application of the NCL model [9] developed by Hearn and Demaine.

Theorem 1.3. *The forklift motion planning problem is PSPACE-complete in the general case.*

The rest of the paper is organized as follows. We give a detailed description of the model of the forklift motion planning problem in Section 2. The configuration graph method is applied to show that the one-box case of the forklift motion planning problem can be solved by a polynomial-time algorithm in Section 3. The general case of the problem is shown to be PSPACE-complete in Section 4. We conclude with a few remarks on future work in Section 5.

2. The Forklift Motion Planning Problem

In this section, we give a formal definition of the Forklift Motion Planning Problem. The problem takes place in a warehouse which can be represented by a finite 3-dimensional[1] grid of size $a \times b \times 2$ (see Figure 2). Each unit cube of the grid can be either unoccupied or occupied by (parts of) the forklift, a box, or a fixed wall.

The forklift occupies 4 cubes in the warehouse. When the fork is in the raised position, the forklift occupies the cubes $(x, y, 1), (x+1, y, 1), (x+1, y, 2)$ and $(x+2, y, 2)$, see Figure 3. When the fork is in the lowered position, the forklift occupies the cubes $(x, y, 1), (x + 1, y, 1), (x + 1, y, 2)$ and $(x + 2, y, 1)$, see Figure 4. Note that there are four different orientations for the forklift, the figures only show the case in which the forklift is facing east.

There are four kinds of possible basic motions for the forklift. The first kind is to lower or to raise the fork of the forklift, either with or without carrying a box. That is switching between the states illustrated in Figure 3 and Figure 4.

The second kind of motion is to move forward or backward. For example in Figure 3, if both of the two cubes $(x + 2, y, 1)$ and $(x + 3, y, 2)$ are unoccupied, then the forklift can move one step forward. Another situation in which the forklift in Figure 3 can move

[1]There is a constant number of vertical layers, so it can be considered as essentially 2-dimensional.

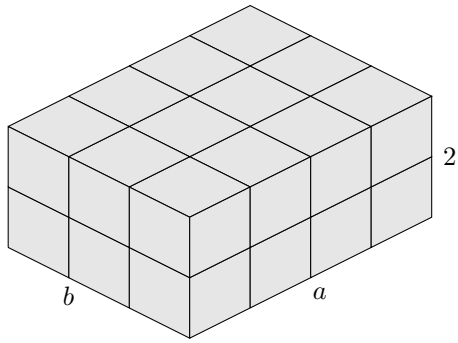

FIGURE 2. The working space

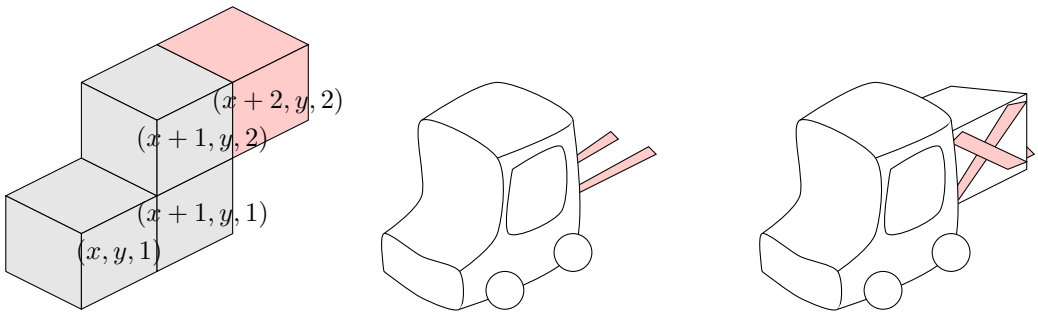

FIGURE 3. The forklift when the fork is raised

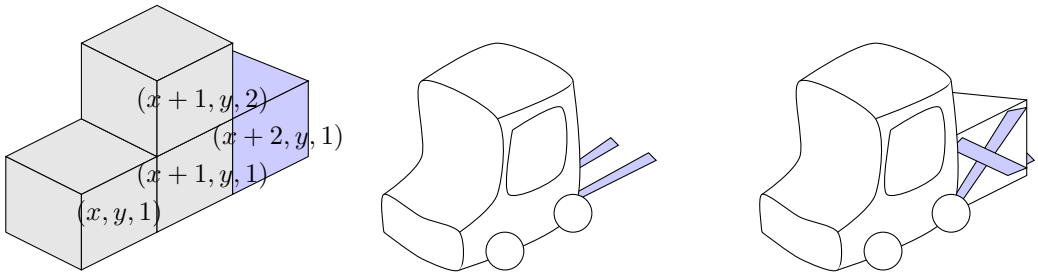

FIGURE 4. The forklift when the fork is lowered

forward is when the cube $(x+2, y, 1)$ is unoccupied and the cube $(x+3, y, 2)$ is occupied by a box (which is on top of a wall of height 1). In this case, the fork goes underneath the box (see also Figure 6a,6b).

The third kind of motion is to turn left or turn right by 90 degrees. Some cubes in the warehouse must be unoccupied to make a turn feasible. In Figure 3, the forklift can make a right turn if and only if the 4 cubes $(x+2, y-1, 2)$, $(x+1, y-1, 2)$, $(x, y+1, 1)$ and $(x+1, y+1, 1)$ are not occupied by boxes or walls. After the turn, the forklift occupies the cubes $(x+1, y+1, 1)$, $(x+1, y, 1)$, $(x+1, y, 2)$ and $(x+1, y-1, 2)$ (see also Figure

6c,6d). The two parts at $(x+1, y, 1)$ and $(x+1, y, 2)$ of the forklift remain in the same locations before and after the turn, so we call them the *central parts* of the forklift.

The fourth kind of motion is to load or unload a box from the fork. This motion can be performed when the fork is either raised or lowered (see Figure 6b,6c). Note that the fork should be facing the box and positioned at the same height as the box before loading it.

In each step, the forklift can perform one of the above four kinds of motion. Also, there are one or more boxes in the warehouse, each of which occupies exactly one cube of the grid. Boxes can only be put down in places with storage marks, either on the floor or on top of a wall of height 1 (see Figure 5). The goal is to transfer all the boxes to the goal positions.

Besides the basic elements of the forklift motion planning problem mentioned above, we introduce yet another element called the *fragile floor*. The forklift can only pass through the fragile floor when it isn't carrying a box. To be more exact, when carrying a box, the central parts cannot be on top of the fragile floor.

To summarize, we give a symbolic, top-down view representation of all the elements of the forklift motion planning problem (Figure 5).

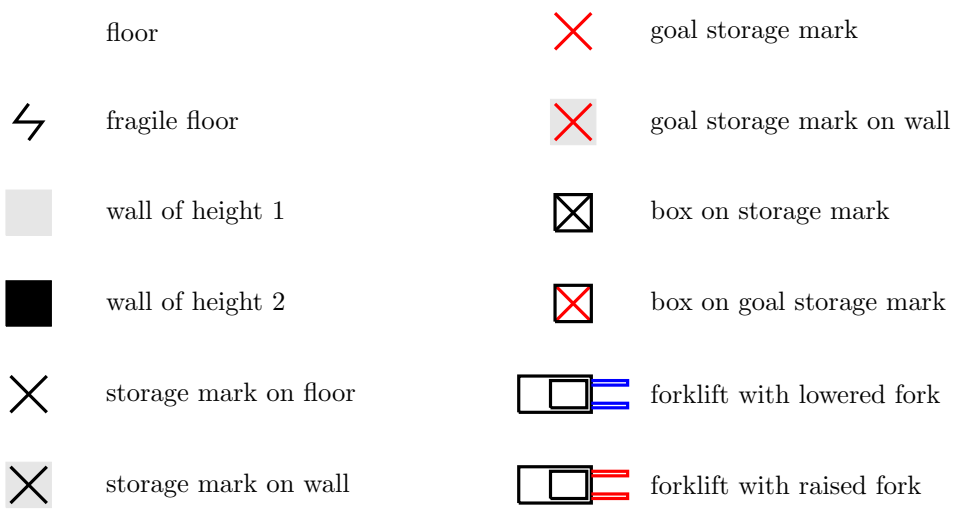

FIGURE 5. The Legend

To conclude this section, we give a concrete example of a sequence of motion performed by the forklift illustrated in Figure 6. Initially, the fork of the forklift is raised (Figure 6a). Then the forklift moves one step forward, which places the fork between the wall of height 1 and the box at height 2 (Figure 6b), and loads the box (Figure 6c). After that, the forklift turns 90 degrees to the right (Figure 6d) and moves one step backward (Figure 6e). Note that the forklift must load the box before making the turn, according to the conditions that allow the forklift to turn right by 90 degrees when the fork is raised.

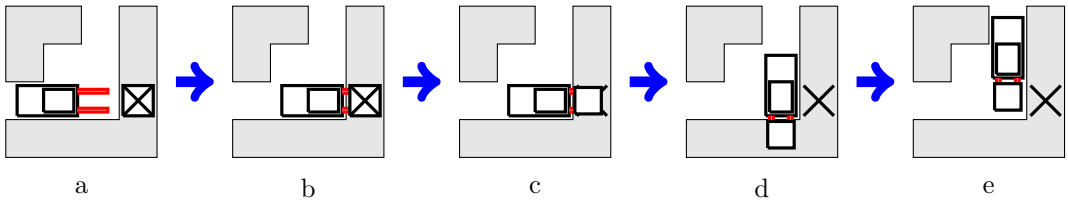

FIGURE 6. Loading a box and turning at the corner

3. The One-Box Case

We prove our first main result in this section. We will prove a simpler case when there are no fragile floors. The proof method can be extended to the case including the fragile floors without much difficulty, because the number of possible configurations remains polynomial.

Proof of Theorem 1.2. To show that the Forklift Motion Planning Problem with one box can be solved by a polynomial algorithm, we construct a configuration graph.

Obviously, in the one-box case, the forklift can just load the box at its initial position and unload it at the goal position, there is no need to put the box down at any moment in between. A configuration during this process can be represented by a 4-tuple (x, y, d, f), where x and y denote the x-coordinate and y-coordinate of the central parts of the forklift, d denotes one of the directions (UP, DOWN, LEFT, RIGHT) the fork of the forklift is facing, and f denotes one of the states (RAISED or LOWERED) of the fork. Each 4-tuple is a vertex of the configuration graph.

Two vertices of the configuration graph are adjacent if and only if the corresponding configurations can be changed from each other by exactly one of the following three basic motions: raising or lowering the fork, moving one step forwards or backwards, and turning left or right. We ignore the fourth kind of motion of loading or unloading a box, because we only need to load the box at the beginning and unload the box in the end for the one-box case. For example, two configurations $(2, 3, \text{UP}, \text{RAISED})$ and $(2, 3, \text{RIGHT}, \text{RAISED})$ are adjacent because they differ by just a left or right turn of the forklift. The configuration graph we construct is a subgraph of the Cartesian product graph $P_a \square P_b \square C_4 \square\ P_2$, where P_k denotes the path of order k and C_4 denotes the cycle of order 4. So the configuration graph has at most $8ab$ vertices.

Let s be the vertex representing the configuration after the forklift loads the box at the initial position, and let t be the vertex representing the configuration of the forklift at the goal position before unloading the box. To find the best sequence of motions for the forklift, we just need to find the shortest path between s and t in the configuration graph. Finding the shortest path between two vertices in an unweighted graph is known to be solvable in polynomial time [10]. Hence, the one-box case for the forklift motion planning problem is also solvable in polynomial time. ∎

4. The General Case

In theory, the configuration graph method used in the one-box case can also be applied to the multiple-box case. However, in the multi-box case, the boxes may block the way

of the forklift such that the boxes need to be transferred to several intermediate positions before finally reaching their goal positions. As a result, there are far more configurations in the multi-box case than in the one-box case. In fact, the number of vertices of the configuration graph may grow exponentially with respect to the number of boxes. This implies that the problem may be harder in the general case. We will prove that it is indeed more difficult and show that the general Forklift Motion Planning Problem is PSPACE-complete by using the NCL (Nondeterministic Constraint Logic) model introduced by Hearn and Demaine [9].

There are several variations of the NCL decision problems, and all of them are PSPACE-complete. For our purpose, we will use the version for the configuration-to-configuration planar NCL.

An instance of the configuration-to-configuration planar NCL problem is defined on a 3-regular planar directed graph, which is called an NCL graph. Each edge of the graph has a weight of either 1 or 2. Moreover, the NCL graph consists of only two kinds of vertices, the AND vertices and the OR vertices. An AND vertex is incident with two edges of weight 1 and one edge of weight 2, and an OR vertex is incident with three edges of weight 2 (Figure 7).

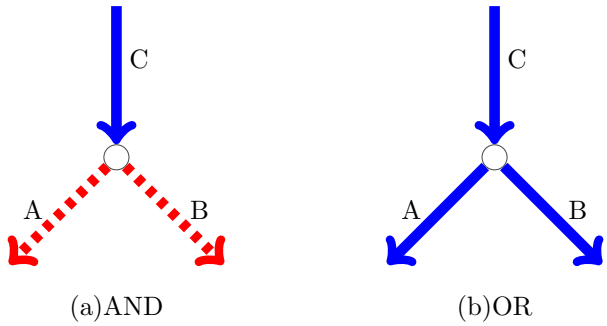

FIGURE 7. NCL (a) an AND vertex, (b) an OR vertex

A configuration of the NCL graph is a specific orientation of the edges, and the configuration is valid if and only if the sum of weights of the incoming edges is at least 2 for each vertex. A valid move for the NCL graph changes the orientation of a single edge, while keeping the configurations valid before and after the change. Given a planar NCL graph, with a valid initial configuration and a valid target configuration for that graph, the NCL problem asks whether there is a sequence of valid moves from the initial configuration to the target configuration such that all intermediate configurations are valid.

Definition 4.1 (NCL Problem)**.**
INPUT. A planar NCL graph with an initial configuration and a target configuration.
OUTPUT. Is there a sequence of valid moves leading the NCL graph from the initial configuration to the target configuration?

Theorem 4.2 ([9])**.** NCL *is PSPACE-complete.*

With the help of the NCL problem, we are ready to prove our second main result.

Proof of Theorem 1.3. Given any instance of the NCL problem, we construct an equivalent instance of the forklift motion planning problem such that the instance of the NCL problem is solvable if and only if the instance of forklift motion planning problem is solvable. Our proof is divided into several subsections. In each of the following subsections, we emulate the edges and vertices of the NCL problem by edge gadgets, vertex gadgets, and several kinds of helper gadgets. These gadgets are put together to form a complete instance of the forklift motion planning problem.

4.1. The Edge Gadget

The edge gadget illustrated in Figure 8 emulates an edge of the NCL problem. It is a half-enclosed area with walls of height 1. The area inside the edge gadget can be of any size or shape, as long as it is large enough for the forklift to turn around. In particular, the edge gadget can turn and split into parallel corridors, as we will need later to form the vertex gadgets. We call the gadget *half-enclosed* because it has two or more doors with fragile floors. With these doors, the forklift can freely get in or out of the internal area of the gadget empty-handed, but cannot get in or out with a box. As a result, the box inside an edge gadget will remain in the gadget. Also, with these doors, the forklift can enter and exit the edge gadget and visit any other part of the warehouse.

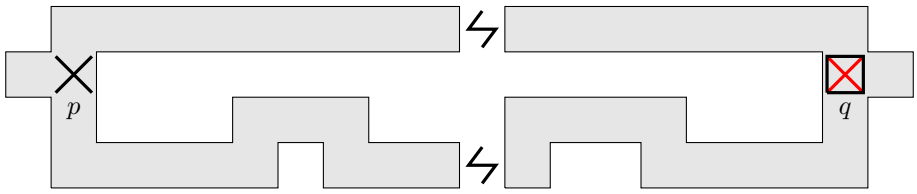

Figure 8. The edge gadget

Inside the edge gadget, there are two storage marks atop the walls of height 1 and one box. Exactly one of the two storage marks is the goal (marked in red), and eventually the box needs to be placed in the goal position. But during the motion planning process, the forklift can always go inside the gadget and change the location of the box between the two marks if the internal passage is not blocked by meet gadgets (see Subsection 4.3). This emulates an edge changing its orientation in the NCL problem. More precisely, the edge gadget illustrated in Figure 8 emulates a directed edge from the vertex q to p. The two storage marks play the role of the two ends of the edge (i.e. linking two vertices), and a box on one of the storage marks means the edge is pointing to the other storage mark.

4.2. The Crossing Gadget

When putting the edge gadgets together, the internal passage of one edge gadget may cross with the internal passage of another gadget. Yet the two edge gadgets need to be independent of each other. In other words, the internal area of two edge gadgets may intersect, but the forklift cannot go from the inside area of one edge gadget to another. This can be easily achieved by the crossing gadget illustrated in Figure 9, where the *crossing gadget* is defined by the area enclosed by the dashed square. Since there is not

enough space for the forklift to turn at the intersecting point between the two edge gadgets A and B, the forklift must remain in the passage of the gadget it initially entered. Many crossing gadgets are needed to form an AND or OR vertex gadget later, so we introduce the symbolic representations of the crossing gadget in Figure 10.

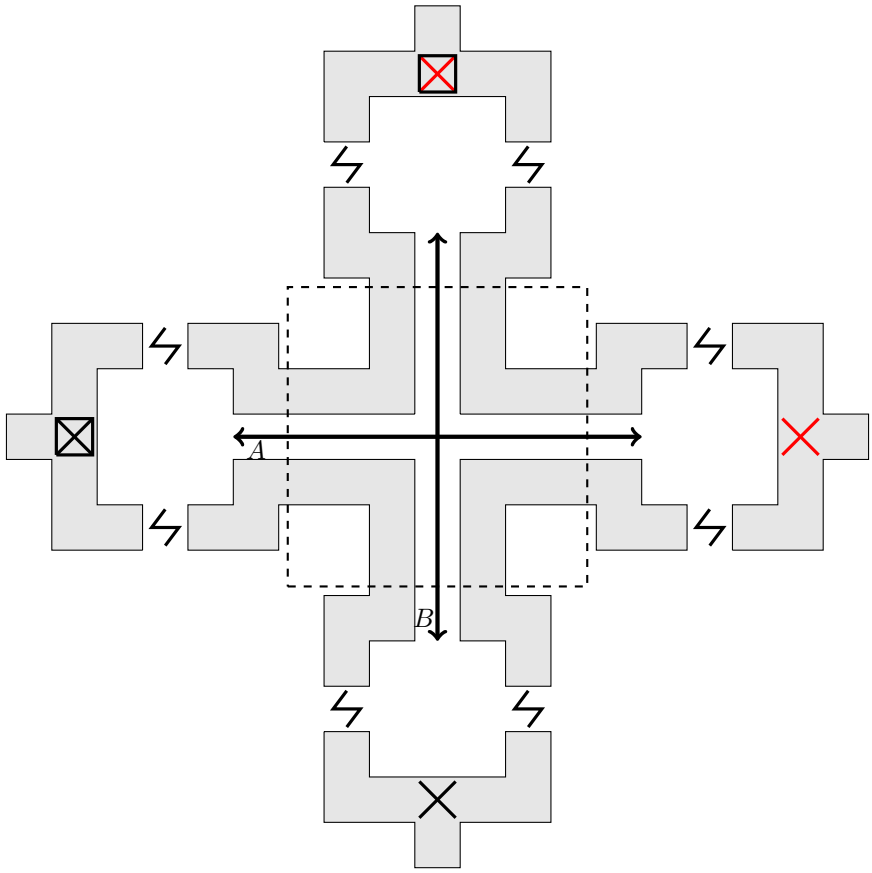

FIGURE 9. The crossing of two gadget A and B

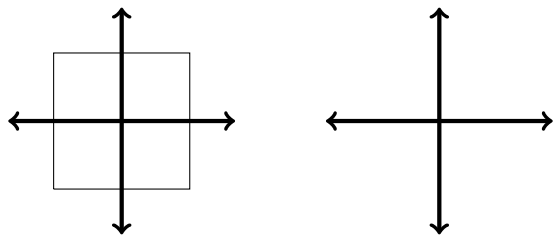

FIGURE 10. The symbolic representations of the crossing gadget

4.3. The Meet Gadget

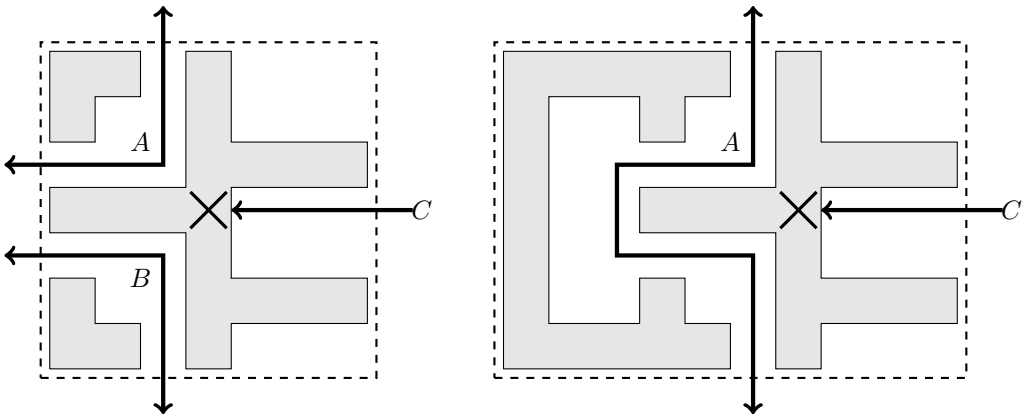

Figure 11. Three edge gadgets A, B and C (left), two edge gadgets (right)

In order to emulate the logic of the AND and OR vertices of the Ncl problem, one key component of our construction method is that the orientation of the edge gadget can block or unblock the internal passages of one or two other edge gadgets, as illustrated in Figure 11. On the left of Figure 11, one end of the edge gadget C meets the internal passages of two other edge gadgets A and B. If the box of the edge gadget C is placed at this end, then the passages A and B are blocked, because there is not enough space for the forklift to turn at the corners (recall the conditions for making a turn in Section 2). We call this a *meet* gadget. A simpler version of the meet gadget is illustrated on the right of Figure 11, where the internal passage of only one edge gadget A can be affected by the existence of a box on storage mark at one end of another edge gadget C. Like the crossing gadgets, we introduce the symbolic representation of the meet gadgets in Figure 12, as we need several of them to form an AND or OR vertex gadget.

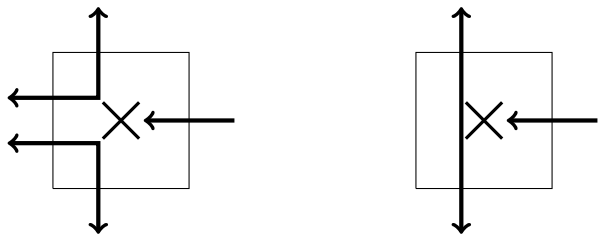

Figure 12. The symbolic representation of two kinds of meet gadgets

4.4. The OR Gadget

Figure 13 shows how three edge gadgets form an OR vertex gadget. Three ends of the edge gadgets (one from each of the three edge gadgets A, B, and C) are grouped to

form the OR vertex gadget. The only constraint of an OR vertex of the NCL problem is that the three edges incident with the OR vertex cannot be pointing away from the vertex at the same time. In Figure 13, the internal area of each edge gadget splits into two parallel passages that later merge to create a single passage leading to the storage mark associated with each edge of the OR vertex gadget. The two parallel passages meet the storage mark of the other two edge gadgets, respectively. So if two edges are pointing away at this vertex, say edges A and B, this means the boxes of edge gadgets A and B are both placed on the storage mark of this end, then the two parallel passages of edge C are both blocked, and there is no way for edge C to point away.

In Figure 13, the OR vertex gadget is formed by the parts enclosed by the dashed rectangle. It roughly includes half of each of the edge gadgets A, B and C. The other ends of the edge gadgets are not shown in this figure, and they will form vertex gadgets with other edge gadgets.

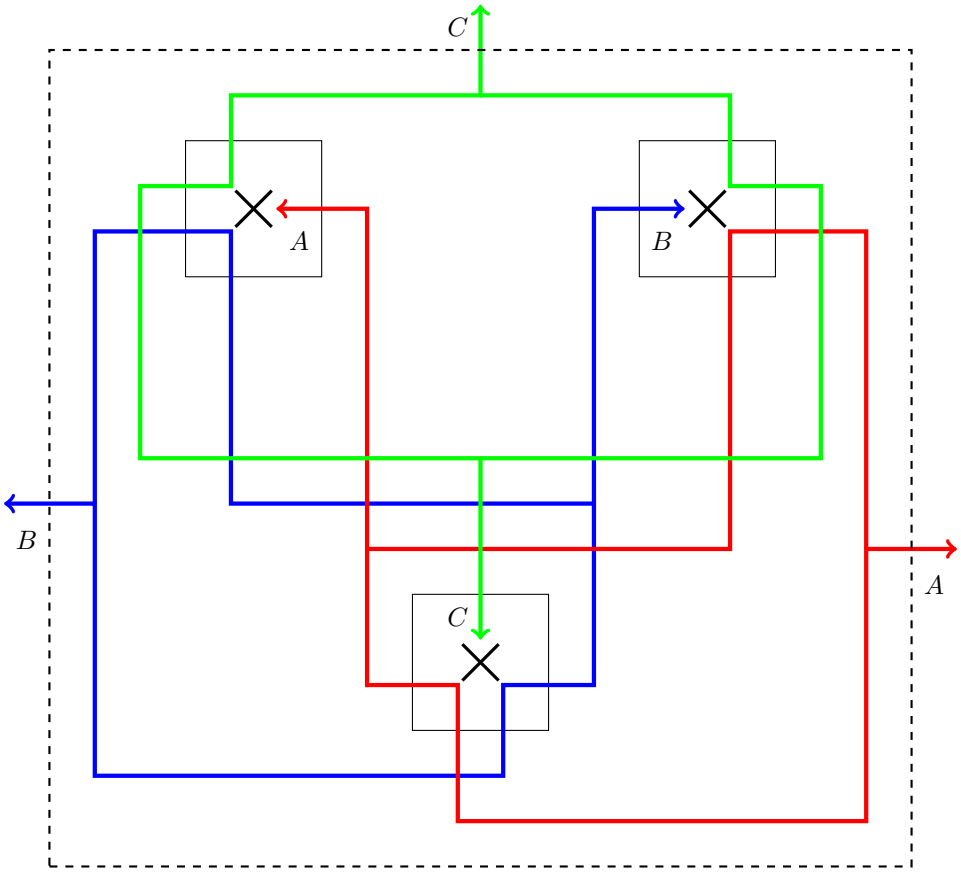

FIGURE 13. The OR gadget

4.5. The AND Gadget

The construction of the AND gadget is illustrated in Figure 14, where three ends from the three edge gadgets are combined differently from the OR gadget. Edge C emulates an edge of weight 2, and edges A and B emulate edges of weight 1. The passage of edge C leading to its storage mark at this end forms meet gadgets with the storage marks of both edges B and C. So if either of the two edges A and B are pointing away (i.e. the storage mark of either edge A or B is occupied by a box), the passage of edge C is blocked, and edge C cannot point away. The passages of edge A and B leading to their respective storage marks both meet the storage mark of the edge C. So if edge C points away (i.e. the box of edge gadget C is on the storage mark of this end, blocking the passages of both edge A and B), neither edge A or B can point away.

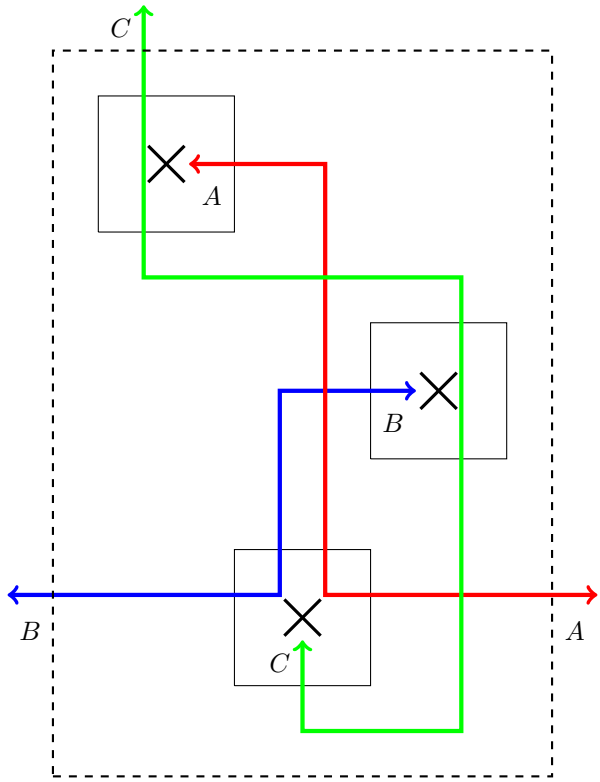

Figure 14. The AND gadget

4.6. Piecing Together

Given an instance of the Ncl problem, we construct an instance of the Forklift problem in the following way, by combining the gadgets described in the previous subsections. For each edge e of the Ncl instance, there is a corresponding edge gadget in the Forklift instance. The goal direction of e determines which of the two storage marks

in the edge gadget is the goal, and the initial direction of e determines on which of the two storage marks the box is initially placed.

For each vertex v of the NCL instance, depending on whether it is an AND or an OR vertex, we just join the ends of the corresponding edge gadgets together to form an AND gadget or an OR gadget. Note that the two kinds of vertex gadget have a fixed size, so the overall size of the FORKLIFT instance is at most polynomial to the size of the NCL instance.

Finally, the forklift can be placed anywhere in the warehouse (if not on top of walls or on top of boxes). As we have mentioned earlier, each edge gadget has two or more doors with fragile floors, so the forklift can visit the internal area of any edge gadget of the warehouse without any trouble and change the orientation of the edge gadget (i.e. transport the box of the edge gadget from one storage mark to the other if the internal passage has not been blocked by meet gadgets) as long as it's not carrying a box while traversing between gadgets. By our design of the vertex gadgets, it is easy to check that the orientation of an edge gadget can be changed if and only if the direction of the corresponding edge in the NCL instance can be changed. Therefore the FORKLIFT instance is solvable if and only if the corresponding NCL instance is solvable.

Since the NCL problem is PSPACE-complete, we have shown that the FORKLIFT problem is PSPACE-hard. To complete the proof, we will show that the Forklift problem is in PSPACE. By Savitch's Theorem, which states that NPSPACE=PSPACE, it suffices to show that the Forklift problem is in NPSPACE. This can be seen by the following nondeterministic polynomial-space algorithm for finding a solution for the Forklift problem. The forklift can nondeterministically traverse the warehouse, at each step nondeterministically choosing a valid move to make, and maintaining the current state but not the previously visited states. The algorithm stops by either finding a solution or exceeding the maximum number of steps needed to find a solution. ∎

5. Conclusion

In this paper, we introduce a 3-dimensional discrete motion planning problem, the FORKLIFT problem. By using a configuration graph, we show that the problem can be solved with a polynomial-time algorithm for the one-box case. In the general case, the problem is PSPACE-complete. But our proof of the PSPACE-completeness makes use of an extra element, the fragile floor. It would be interesting to determine the complexity class of the forklift motion planning problem without fragile floors in future research.

Acknowledgements

The author would like to thank the anonymous referees for their very valuable comments and suggestions which help to improve this paper. This work was supported by the Research Fund of Guangdong University of Foreign Studies (Nos. 297-ZW200011 and 297-ZW230018), and the National Natural Science Foundation of China (No. 61976104).

References

[1] L.E. Kavraki, S.M. LaValle, Motion Planning, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2016.

[2] J.F. Canny, Complexity of Robot Motion Planning, MIT Press, 1988.

[3] G.W. Flake, E.B. Baum, Rush Hour is PSPACE-complete, or why you should generously tip parking lot attendants, Theoretical Computer Science 270 (1) (2002) 895–911.

[4] J.R. Hartline, R. Libeskind-Hadas, The computational complexity of motion planning, SIAM Review 45 (3) (2003) 543–557.

[5] K. Buchin, M. Buchin, Rolling block mazes are PSPACE-complete, Journal of Information Processing 20 (3) (2012) 719–722.

[6] B. Meyer, Generalized Pete's Pike is PSPACE-complete, Theoretical Computer Science 613 (2016) 115–125.

[7] W. He, Z. Liu, C. Yang, Snowman is PSPACE-complete, Theoretical Computer Science 677 (2017) 31–40.

[8] J. Brunner, L. Chung, E.D. Demaine, D. Hendrickson, A. Hesterberg, A. Suhl, A. Zeff, 1×1 rush hour with fixed blocks is PSPACE-complete, in 10th International Conference on Fun with Algorithms (FUN 2021) (2020) 7:1–7:14.

[9] R.A. Hearn, E.D. Demaine, PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation, Theoretical Computer Science 343 (1) (2005) 72–96.

[10] D. West, An Introduction to Graph Theory, Second edition, Prentice Hall, 2001.